CMPT 270
Midterm 1

55 minutes — Closed Book — Nov. 9, 2007

Marks — Java syntax sheets allowed

7 1. Give cases that should be tested when testing a method to insert a new value into an ordered list that is implemented using an array.

6 2. In class, the value of decomposing an application into layers was discussed. In particular, the bank application was divided into the following layers: interface, controller, command, and data storage.
   (a) Why is an application partitioned into layers?
   (b) Two guidelines in doing decomposition were coupling and cohesion. How does coupling relate to how the application is partitioned?

5 3. To ensure that an object is always in a valid state, a representation invariant function (e.g. boolean repOk( )) can be defined to test that the state of the current object is valid. Suppose that a class is meant to model a triangle, and the representation stores the lengths of the three sides. What specific tests should the representation invariant function do?

4 4. One of the types of abstraction is procedural abstraction. A key part of procedural abstraction is having a precise specification of each procedure/method that is independent of its implementation. What are the parts of the precise specification of a procedure?

12 5. Suppose that there is an interface A, and two classes B and C, which are as simple possible subject to the following:
   - B is an implementation of A
   - C has a field called *a* of type A
   - C has a constructor with a local variable *b* of type B
   - The constructor of C has the statement *a.r*( ), that during execution invokes a method *r*( ) defined in class B. Note that this requirement implies the existence of other parts that must be included.

   (a) Give the UML class diagram for the above situation and what it implies (or at least as much of it as can be portrayed in a class diagram).
   (b) Give the Java code for A, B and C. Keep them as simple as possible. In particular, the method *r* might as well have no parameters and an empty body.

16 6. Suppose that a class Person exists, and it has a constructor with one parameter of type String. The objective of this question is to define a descendant of Person called Employee. This class should have an addition field called *number* of type **int**. The class should have a constructor, and the two methods equals and toString originally defined in the class Object. The equals method should compare two Employees by comparing their numbers. Recall that since equals was originally defined in class Object, its parameter has type Object. The toString method should append the number of the Employee onto the value obtained from invoking toString of the Person class. The class need not have any other fields or methods. Give the code for the class Employee. Do not give the code for the class Person.

Total 50 — The end